What is claimed is:

[c1] 1.A method for efficient processing of tasks in a communications system, comprising:

sampling a current task identifier and a next task identifier;

providing a first register file for storing values for a current task;

providing a second register file for storing values for the current task that are not in the first register file;

providing a third register file for preloading values for the next task; and

performing a task switch by making the next task identifier the current task identifier and sampling a further next task identifier.

[c2] 2.The method of claim 1, further comprising the step of completing the preload of the register values for the next task identifier which after the task switch is the current task identifier.

[c3] 3.The method of claim 2, further comprising using the third register file as the second register`file after the task switch,.

[c4] 4.The method of claim 1, wherein the first register file comprises registers with a data field and a task identifier field.

[c5] 5.The method of claim 4, wherein the first register file has 32 registers, each register having a 32 bit data field and a 6 bit task identifier field.

[c6] 6.The method of claim 1, wherein the first register file is exposed to a programmer of the communications processor and the second register file and the third register file are hidden from the programmer.

[c7] 7.The method of claim 1, wherein task switches are performed without an explicit save/restore of the register files.

[c8] 8.The method of claim 1, further comprising performing a write during execution of the current task by:

comparing the current task identifier to a task identifier in the first register file;

writing a value to the first register file when the current task identifier is the same as the task identifier in the first register file; and writing a value to the first register file when the current task identifier is not the same as the task identifier in the first register file after the content in the first register file is saved to a memory.

[c9] 9.The method of claim 8, wherein the content in the first register file is saved to a task identifier context table.

[c10] 10.The method of claim 1, further comprising performing a read during execution of the current task by:
comparing the current task identifier to a task identifier in the first register file;
reading a value from the first register file when the current task identifier is the same as the task identifier in the first register file; and
reading a value from the second register file when the current task identifier is not the same as the task identifier in the first register file.

[c11] 11.The method of claim 10, wherein the content of the first register file is not changed as a result of the read.

[c12] 12.A system for efficient processing of tasks in a communications system, comprising:
means for sampling a current task identifier and a next task identifier;
a first register file for storing values for a current task;
a second register file for storing values for the current task that are not in the first register file;
a third register file for preloading values for the next task; and
means for performing a task switch by making the next task identifier the current task identifier and sampling a further next task identifier.

[c13] 13.The system of claim 12, wherein the means for performing a task switch completes the preload of the register values for the next task identifier which after the task switch is the current task identifier.

[c14] 14.The system of claim 12, wherein the means for performing a task switch uses the third register file as the second register file after the task switch.

[c15]

15.The system of claim 12, wherein the first register file comprises registers with a data field and a task identifier field.

[c16] 16.The system of claim 15, wherein the first register file has 32 registers, each register having a 32 bit data field and a 6 bit task identifier field, and further wherein the second register file and the third register file each have 32 registers.

[c17] 17.The system of claim 12, further comprising a processor which performs a write during execution of the current task by:
comparing the current task identifier to a task identifier in the first register file;
writing a value to the first register file when the current task identifier is the same as the task identifier in the first register file; and
writing a value to the first register file when the current task identifier is not the same as the task identifier in the first register file after the content in the first register file is saved to a memory.

[c18] 18.The system of claim 17, wherein the content in the first register file is saved to a task identifier context table.

[c19] 19.The system of claim 12, further comprising a processor which performs a read during execution of the current task by:
comparing the current task identifier to a task identifier in the first register file;
reading a value from the first register file when the current task identifier is the same as the task identifier in the first register file; and
reading a value from the second register file when the current task identifier is not the same as the task identifier in the first register file.

[c20] 20.The system of claim 19, wherein the content of the first register file is not changed as a result of the read.

[c21] 21.The system of claim 12, wherein the means for performing a task switch comprises a preload and bump unit.

[c22] 22.The system of claim 17, wherein the processor is an ALU.

[c23] 23.The system of claim 19, wherein the processor is an ALU.